# Streaming Graph Partitioning in the Planted Partition Model

Charalampos E. Tsourakakis
Harvard School of Engineering and Applied Sciences
babis@seas.harvard.edu

## ABSTRACT

The sheer increase in the size of graph data has created a lot of interest into developing efficient distributed graph processing frameworks. Popular existing frameworks such as GraphLab and Pregel rely on balanced graph partitioning in order to minimize communication and achieve work balance.

In this work we contribute to the recent research line of streaming graph partitioning [30, 31, 34] which computes an approximately balanced $k$-partitioning of the vertex set of a graph using a single pass over the graph stream using degree-based criteria. This graph partitioning framework is well tailored to processing large-scale and dynamic graphs. In this work we introduce the use of higher length walks for streaming graph partitioning and show that their use incurs a minor computational cost which can significantly improve the quality of the graph partition. We perform an average case analysis of our algorithm using the planted partition model [7, 25]. We complement the recent results of Stanton [30] by showing that our proposed method recovers the true partition with high probability even when the gap of the model tends to zero as the size of the graph grows. Furthermore, among the wide number of choices for the length of the walks we show that the proposed length is optimal. Finally, we perform simulations which indicate that our asymptotic results hold even for small graph sizes.

## Categories and Subject Descriptors

G.2.2 [**Graph Theory**]: Graph Algorithms

## General Terms

Theory, Experimentation

## Keywords

Streaming Graph Partitioning; Planted Partition Model; Distributed Computing


*COSN'15,* November 2–3, 2015, Palo Alto, California, USA.

DOI: http://dx.doi.org/10.1145/2817946.2817950.

## 1. INTRODUCTION

The size of graph data that are required to be processed nowadays is massive. For instance, the Web graph amounts to at least one trillion of links [1] and Facebook in 2012 reported more than 1 billion of users and 140 billion of friend connections. Furthermore, graphs of significantly greater size emerge by post- processing various other data such as image and text datasets. This sheer increase in the size of graphs has created a lot of interest in developing distributed graph processing systems [15, 21, 23], in which the graph is distributed across multiple machines. A key problem towards enabling efficient graph computations in such systems is the **NP**-hard problem of *balanced graph partitioning*. High-quality partitions ensure low volume of communication and work balance.

Recently, Stanton and Kliot [31] introduced a streaming graph partitioning model. This line of research despite being recent and lacking theoretical understanding has already attracted a lot of interest. Several existing systems have incorporated this model such as PowerGraph [13]. The framework of Fennel has been adapted by PowerLyra [6], which has been included into the most recent GraphLab version [3, 21] yielding significant speedups for various iterative computations. Stanton performed an average case analysis of two streaming algorithms, and explained their efficiency despite the pessimistic worst case analysis [30]. Despite the fact that existing established heuristics such as Metis typically outperform streaming algorithms, the latter are well tailored to today's needs for processing dynamic graphs and big graph data which do not fit in the main memory. They are computationally cheap and provide partitions of comparable quality. For instance, Fennel on the Twitter network with more than 1.4 billion edges performed comparably well with Metis for a wide variety of settings, requiring 40 minutes of running time, whereas Metis $8\frac{1}{2}$ hours.

So far, the work on streaming graph partitioning is based on computing the degrees of incoming vertices towards each of the $k$ available machines. Equivalently, this can be seen as performing a one step random walk from the incoming vertex. A natural idea which is used extensively in the literature of graph partitioning [20, 29, 37] is the use of higher length walks. In this work we introduce this idea in the setting of streaming graph partitioning. At the same time we maintain the time efficiency of streaming graph partitioning algorithms which make them attractive to various graph processing systems [13, 15, 21, 23].

**Summary of our contributions**. Our contributions can be summarized as follows:

- Our proposed algorithm introduces the idea of using higher length walks for streaming graph partitioning. It incurs a negligible computational cost and significantly improves the quality of the partition. We perform an average case analysis on the planted partition model, c.f. Section 3.2 for the description of the model. We complement the recent results of [30] which require that the gap $p-q$ of the planted partition model is constant in order to recover the partition *whp*[1], by allowing gaps $p-q$ which asymptotically tend to 0 as $n$ grows.
- Among the wide number of choices for the length of $t$-walks where $t \geq 2$, we show that walks of length 2 are optimal as they allow the smallest possible gap for which we can guarantee recovery *whp*, c.f. Section 3.5.
- We evaluate our method on the planted partition model and we provide simulation results that strongly indicate that our asymptotic results hold even for small graph sizes.
- We complement prior work by showing that Fennel's optimal quasi-clique partitioning objective [34, 33] is NP-hard.

## 2. RELATED WORK

**Balanced graph partitioning.** The *balanced graph partitioning* problem is a classic **NP**-hard problem of fundamental importance to parallel and distributed computing. The input to this problem is an undirected graph $G(V,E)$ and an integer $k \in \mathbb{Z}^+$, the output is a partition of the vertex set in $k$ balanced sets such that the number of edges across the clusters is minimized. We refer to the $k$ sets as *clusters* or *machines* interchangeably. Formally, the balance constraint is defined by the imbalance parameter $\nu$. Specifically, the $(k,\nu)$-balanced graph partitioning asks to divide the vertices of a graph in $k$ clusters each of size at most $\nu\frac{n}{k}$, where $n$ is the number of vertices in $G$. The case $k=2, \nu=1$ is equivalent to the **NP**-hard minimum bisection problem. Several approximation algorithms, e.g., [9], and heuristics, e.g., [16], exist for this problem. When $\nu = 1+\epsilon$ for any desired but fixed $\epsilon > 0$ there exists a $O(\epsilon^{-2}\log^{1.5} n)$ approximation algorithm [18]. When $\nu = 2$ there exists an $O(\sqrt{\log k \log n})$ approximation algorithm based on semidefinite programming (SDP) [19]. Due to the practical importance of $k$-partitioning there exist several heuristics, among which Metis [28] stands out for its good performance. Survey [5] summarizes many popular existing methods for the balanced graph partitioning problem.

**Streaming balanced graph partitioning.** Despite the large amount of work on the balanced graph partitioning problem, neither state-of-the-art approximation algorithms nor heuristics such as Metis are well tailored to the computational restrictions that the size of today's graphs impose.

Stanton and Kliot introduced streaming balanced graph partitioning. In this setting the graph arrives as a stream and decisions about the partition must be taken online [31]. Specifically, when a vertex with its neighbors arrives, the partitioner decides where to place the vertex "on the fly", using limited computational resources (time and space). A vertex is never relocated after it becomes assigned to one of the $k$ machines. Fennel generalized the notion of optimal quasi-cliques [33] to $k$-partitioning and is closely related to detecting large near-cliques [12, 32]. This extension provided well-performing decision strategies for streaming graph partitioning. In the Appendix we prove that Fennel is NP-hard. Both [31] and [34] can be adapted to edge streams. Nishimura and Ugander [27] consider a variation of Fennel that allows multiple passes over the stream. It is worth mentioning that very recently Margo and Seltzer provided a state-of-art distributed streaming graph partitioner [24]. Stanton showed that streaming graph partitioning algorithms with a single pass even for random stream orders cannot approximate the optimal cut size within $o(n)$ [30]. Stanton [30] analyzes two variants of well performing algorithms on real-data from [31] on random graphs. Specifically, Stanton proves that if the graph $G$ is sampled according to the planted partition model, then the two algorithms despite their similarity may perform differently. Furthermore, one of the two recovers the true partition *whp*, assuming that inter-, intra-cluster edge probabilities are constant, and their gap is constant.

**Planted partition model.** Jerrum and Sorkin [14] studied the planted bisection model, a random undirected graph with an even number of vertices. According to this model, each half of the vertices is assigned to one of two clusters. Then, the probability of an edge $(i,j)$ is $p$ if $i,j$ have the same color, otherwise $q < p$. We will refer to $p,q$ as the intra- and inter-cluster probabilities. Their difference $p-q$ will be referred to as the *gap* of the model. Condon and Karp [7] studied the generalization of the planted bisection problem where instead of having only two clusters, there exist $k$ clusters of equal size. The probability of an edge is the same as in the planted bisection problem. They show that the hidden partition can be recovered *whp* if the gap satisfies $p-q \geq n^{-1/2+\epsilon}$. McSherry [25] presents a spectral algorithm that recovers the hidden partition in a random graph *whp* if $p-q = \Omega(n^{-1/2+\epsilon})$. Recently, Van Vu gave an alternative algorithm [36] to obtain McSherry's result. Zhou and Woodruff [37] showed that if $p = \Theta(1), q = \Theta(1), p-q = \Omega(n^{-1/4})$, then a simple algorithm based on squaring the adjacency matrix of the graph recovers the hidden partition *whp*.

**Random walks.** The idea of using walks of length greater than one [37] is common in the general setting of graph partitioning. Lovász and Simonovits [20] show that random walks of length $O(\frac{1}{\phi})$ can be used to compute a cut with sparsity at most $\tilde{O}(\sqrt{\phi})$ if the sparsest cut has conductance $\phi$. Later, Spielman and Teng [29] provided a local graph partitioning algorithm which implements efficiently the Lovász-Simonovits idea.

**Theoretical preliminaries.**[2] A useful lemma that we use extensively is Boole's inequality, also known as the union bound.

Lemma 1 (Union bound). *Let $A_1,\ldots,A_n$ be events in a probability space $\Omega$, then*

$$\mathbf{Pr}\left[\cup_{i=1}^n A_i\right] \leq \sum_{i=1}^n \mathbf{Pr}\left[A_i\right].$$

The following theorem is due to Markov and its use is known as the first moment method.

[1]An event $\mathcal{E}_n$ occurs *with high probability*, or *whp* for brevity, if $\lim_{n\to\infty} \mathbf{Pr}\left[\mathcal{E}_n\right] = 1$.

[2] A detailed description can be found in [8, 26].

Theorem 1 (First Moment Method). *Let $X$ be a non-negative random variable with finite expected value $\mathbb{E}[X]$. Then, for any real number $t > 0$,*

$$\mathbf{Pr}[X \geq t] \leq \frac{\mathbb{E}[X]}{t}.$$

The following theorem is due to Chebyshev and its use is known as the second moment method.

Theorem 2 (Second Moment Method). *Let $X$ be a random variable with finite expected value $\mathbb{E}[X]$ and finite non-zero variance $\mathbb{V}ar[X]$. Then, for any real number $t > 0$,*

$$\mathbf{Pr}[|X - \mathbb{E}[X]| \geq t] \leq \frac{\mathbb{V}ar[X]}{t^2}.$$

Finally, we use the following Chernoff bounds for independent and negatively correlated random variables.

Theorem 3 (Multiplicative Chernoff Bound). *Let $X = \sum_{i=1}^{n} X_i$ where $X_1, \ldots, X_n$ are independent random variables taking values in $[0, 1]$. Also, let $\delta \in [0, 1]$. Then,*

$$Pr(X \leq (1-\delta)E(X)) \leq e^{-\delta^2 E(X)/2}.$$

Definition 1. *Let $X_1, \ldots, X_n$ be random binary variables. We say that they are negatively correlated if and only if for all sets $I \subseteq [n]$ the following inequalities are true:*

$$Pr(\forall i \in I : X_i = 0) \leq \prod_{i \in I} Pr(X_i = 0)$$

*and*

$$Pr(\forall i \in I : X_i = 1) \leq \prod_{i \in I} Pr(X_i = 1).$$

Theorem 4. *Let $X = \sum_{i=1}^{n} X_i$ where $X_1, \ldots, X_n$ are negatively correlated binary random variables. Also, let $\delta \in [0, 1]$. Then,*

$$Pr(X \leq (1-\delta)E(X)) \leq e^{-\delta^2 E(X)/2}.$$

# 3. PROPOSED ALGORITHM

**Section outline.** Section 3.1 introduces our notation and Section 3.2 presents in detail the random graph model we analyze. Section 3.3 provides two useful lemmas used in Sections 3.4 and 3.5. Section 3.4 shows an efficient way to recover the true partition of the planted partition model *whp* using walks of length $t = 2$ even when $p - q = o(1)$, i.e., the gap is asymptotically equal to 0. Section 3.5 shows that when the length of the walk $t$ is set to 2, then we obtain the smallest possible *gap* $p - q$ for which we can guarantee recovery of the partition *whp*. We do not try to optimize constants in our proofs, since we are interested in asymptotics. Our proofs use the elementary inequalities $1 - p \leq e^{-p}$, $\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$ and the probabilistic tools presented in the previous Section.

## 3.1 Notation

Let $G(V, E)$ be a simple undirected graph, where $|V| = n, |E| = m$. We define a *vertex partition* $\mathcal{P} = (S_1, \ldots, S_k)$ as a family of pairwise disjoint vertex sets whose union is $V$. Each set $S_i$ is assigned to one of $k$ machines, $i = 1, \ldots, k$. We refer to each $S_i$ as a *cluster* or *machine*. Throughout this work, we assume $k = \Theta(1)$. Let $\partial e(\mathcal{P})$ be the set of edges that cross partition boundaries, i.e., $\partial e(\mathcal{P}) = \cup_{i=1}^{k} e(S_i, V \setminus S_i)$. The fraction of edges cut $\lambda$ is defined as $\lambda = \frac{|\partial e(\mathcal{P})|}{m}$. The imbalance factor or normalized maximum load $\rho$ is defined as $\rho = \max_{1 \leq i \leq k} \frac{|S_i|}{\frac{n}{k}}$. Notice that $\rho$ satisfies the double inequality $1 \leq \rho \leq k$. When $\rho = 1$ we have a perfectly balanced partition. At the other extreme, when $\rho = k$ all the vertices are placed in one cluster, leaving $k - 1$ clusters empty. In practice, there is a constraint $\rho \leq \nu$ where $\nu$ is a value imposed by application-specific restrictions. Typically, $\nu$ is close to 1. We omit floor and ceiling notation for simplicity, our results remain valid.

## 3.2 Planted Partition Model

The model $G(n, \Psi, P)$ is a generalization of the classic Erdös-Rényi graph [4]. The first parameter $n$ is the number of vertices. The second parameter of the model is the function $\Psi : [n] \to [k]$ which maps each vertex to one of $k$ clusters. We refer to $C_i = \Psi^{-1}(i)$ as the $i$-th cluster, $i = 1, \ldots, k$. The third parameter $P$ is a $k \times k$ matrix such that $0 \leq P_{ij} \leq 1$ for all $i, j = 1, \ldots, k$ which specifies the probability distribution over the edge set. Specifically, a graph $G \sim G(n, \Psi, P)$ is generated by adding an edge with probability $P(\Psi(u), \Psi(v))$ between each pair of vertices $\{u, v\}$. Notice, that when all the entries of $P$ are equal to $p$, then $G(n, \Psi, P)$ is equivalent to the $G(n, p)$ model. In this work we are interested into graphs that exhibit clustering. The planted partition model we analyze is the same as in [7, 25, 30, 37]: $P_{ij} = p\mathbf{1}(i = j) + q\mathbf{1}((i \neq j))$ for all $i, j = 1, \ldots, k$. We assume that $p > q = \Theta(1)$. For simplicity, we refer to this version of $G(n, \Psi, P)$ as $G(n, k, p, q)$.

## 3.3 Results and useful lemmas

We prove two simple lemmas that we use in the analysis of our algorithms. We refer to the vertex that arrives exactly after $i - 1$ vertices as the $i$-th vertex.

Lemma 2. *For all $i \geq D \log^{1+\delta}(n)$ where $D, \delta > 0$ are any positive constants, there exist at least $\frac{i}{2k}$ vertices that have already arrived from each cluster $j$ whp, $j = 1, \ldots, k$.*

Proof. Fix any index $i \geq D \log^{1+\delta}(n)$. Define for each $j \in [k]$ the bad event $\mathcal{A}_j$ that cluster $j$ has less than $\frac{i}{2k}$ vertices after the arrival of $i$ vertices. Given our assumption on the random order of the stream, the first $i$ vertices form a random $i$-subset of $[n]$. Let $Y_j$ be the number of vertices from cluster $j$ among the first $i$ vertices. We observe that the distribution of $Y_j$ is the hypergeometric distribution $H(n, n/k, i)$[3]. Therefore, we obtain the following exact expression for the probability of the bad event $\mathcal{A}_j$:

[3]Recall, the hypergeometric distribution $H(N, R, s)$ is a discrete probability distribution that describes the probability of choosing $r$ red balls in $s$ draws of balls without replacement from a finite population of size $N$ containing exactly $R$ red balls.

$$\mathbf{Pr}\left[\mathcal{A}_j\right] = \sum_{r=0}^{i/2k} \mathbf{Pr}\left[Y_j = r\right] = \sum_{r=0}^{i/2k} \frac{\binom{n/k}{r}\binom{n(k-1)/k}{i-r}}{\binom{n}{i}}.$$

Even if an asymptotic analysis using Stirling's formula is possible, a less tedious approach is possible. We express $Y_j$ as the sum of $i$ indicator variables $Y_{j,1}, \ldots, Y_{j,i}$ where $Y_{j,l} = 1$ if and only if the $l$-th vertex $v$ has $\Psi(v) = j$, $l = 1, \ldots, i$. Clearly, $\mathbb{E}\left[Y_j\right] = \frac{i}{k}$. Notice that even if the indicator random variables are not independent, they are negatively correlated. Therefore, Theorem 4 applies, obtaining

$$\mathbf{Pr}\left[Y_j \leq (1 - \frac{1}{2})\frac{i}{k}\right] \leq e^{-\frac{i}{8k}} \leq e^{-\frac{D\log^{1+\delta}(n)}{8k}} \ll o(n^{-1}).$$

The proof is completed by taking a union bound over $k$ machines and $(n - D\log^{1+\delta}(n))$ vertices. Specifically, let $\mathcal{E}$ be the event that there exists an index $i \geq D\log^{1+\delta}(n)$ such that $\cup_{j=1}^{k} \mathcal{A}_j$ is true. Then, $\mathbf{Pr}\left[\mathcal{E}\right] \leq (n - D\log^{1+\delta}(n))ko(n^{-1}) = o(1)$. □

Lemma 3. *Let $\delta > 0$ be any positive constant. After $\log^{6+\delta}(n)$ vertices have arrived, all remaining vertices in the stream have $\log^6(n)$ neighbors which reside in the $k$ machines.*

Proof. Let $i = g(n) \geq \log^{6+\delta}(n)$, be the number of vertices that have arrived in the incidence stream. Let $\mathcal{E}$ be the event that there exists a vertex $v$ that has arrived after the first $i$ vertices with less than $\log^6(n)$ neighbors in the set of arrived vertices. We obtain that $\mathbf{Pr}\left[\mathcal{E}\right] = o(1)$ using the following upper bound.

$$\begin{aligned}
\mathbf{Pr}\left[\mathcal{E}\right] &\leq n \sum_{j=0}^{\log^6(n)} \binom{g(n)}{j} \left(\frac{p}{1-q}\right)^i (1-q)^{g(n)} \\
&\leq n \sum_{j=0}^{\log^6(n)} \left(\frac{g(n)e}{j}\right)^j \left(\frac{p}{1-q}\right)^j (1-q)^{g(n)} \\
&\leq Cn\log^6(n)\Big(\frac{pg(n)e}{j(1-q)}\Big)^{\log^6(n)} e^{-qg(n)} = o(1).
\end{aligned}$$

□

## 3.4 Path-2 classification

The main theoretical result of this Section is that we can use paths of length 2 to recover the partition $\Psi$ *whp* even when $p - q = o(1)$. Our algorithm avoids making final decisions for the first $B$ vertices until the end of the process and uses the fact that vertices from the same cluster have more common neighbors compared to a pair of vertices from different clusters. Our algorithm is shown below.

1. Place the first $B = \log^{6+\delta}(n)$ vertices in any of the $k$ machines, marked as non-classified. Here, $\delta > 0$ is any positive constant.
2. Let $S$ be a random sample of size $3k\log n$ vertices from the set of $B$ non-classified vertices.
3. Let $R$ be a random sample of $\log^6 n$ vertices from the set of $B$ non-classified vertices.
4. For the $j$-th vertex, $B + 1 \leq j \leq n$, do the following:
   - For each $x \in S$ compute the number of common neighbors of $j, x$ in $R$.
   - Let $M = (p^2 + (k-1)q^2)\log^6 n$. Assign $j$ to the same cluster with a vertex $x^* \in S$ which has at least $M - M^{2/3}$ common neighbors with $j$. Ties are always assigned to the vertex with the smallest id. Remove non-classified tag from $x^*$.
5. Perform the same procedure for the remaining, if any, non-classified vertices.

We prove the correctness of the algorithm. The next lemma states that when we obtain the random sample $S$, there always exists at least one vertex from each cluster of the partition. This is critical since our algorithm assigns each incoming vertex $v$ to a representative vertex from cluster $\Psi(v)$. Among the various possible choices for a representative of cluster $c$, we choose the vertex $u$ with the minimum vertex id, namely $u = \arg\min\{u \in S : \Psi(u) = c\}$.

Lemma 4. *Let $S$ be a random sample of size $3k\log n$ vertices from a population of $j \geq \log^{6+\delta}(n)$ vertices. Then, with high probability there exists at least one representative vertex from each cluster of the planted partition in $S$*

Proof. First, notice that by Lemma 2 there exist at least $j/2k \geq \frac{\log^{6+\delta}(n)}{2k}$ vertices from each cluster $i = 1, \ldots, k$. Let $\mathcal{E}_i$ be the event of failing to sample at least one vertex from cluster $i$ of the partition, $i = 1, \ldots, k$. We can upper bound the probability of the union $\cup_{i=1}^{k} \mathcal{E}_i$ of these bad events as follows:

$$\begin{aligned}
\mathbf{Pr}\left[\cup_{i=1}^{k} \mathcal{E}_i\right] &\leq k\Big(1 - \frac{\frac{j}{2k}}{j}\Big)^{3k\log n} \\
&\leq ke^{-3k\log n/2k} = o(n^{-1}).
\end{aligned}$$

□

The next theorem is our main theoretical result. It states that the algorithm recovers the true partition $\Psi$ *whp*.

Lemma 5. *If $p = \Theta(1), q = \Theta(1), p - q = \omega\big(\frac{1}{\log(n)}\big)$, then all vertices are classified correctly whp. The algorithm runs in sublinear time.*

Proof. Let $j$ be the index of the incoming vertex, $x \in S$. We condition on the event $\mathcal{E}$ that there exists at least one vertex from each cluster of the planted partition in the sample $S$. Define $Y_x$ as the number of triples $(j, u, x)$ where $u \in R$ and $x \in S$ such that $\Psi(j) = \Psi(x)$. Similarly, let $Z_x$ be the number of triples $(j, u, x)$, where $u \in R$ and $x \in S$

such that $\Psi(j) \neq \Psi(x)$. The expected values of $Y_x, Z_x$ are respectively

$$\mathbb{E}[Y_x] = \left(p^2 + (k-1)q^2\right) \log^6 n,$$

and

$$\mathbb{E}[Z_x] = \left(2pq + (k-2)q^2\right) \log^6 n.$$

A direct application of the multiplicative Chernoff bound, see Theorem 3, yields

$$\mathbf{Pr}\left[Y_x \leq \mathbb{E}[Y_x] - \mathbb{E}[Y_x]^{2/3}\right] \leq e^{-O(\mathbb{E}[Y_x]^{1/3})} = n^{-O(\log n)}$$

and

$$\mathbf{Pr}\left[Z_x \geq \mathbb{E}[Z_x] + \mathbb{E}[Z_x]^{2/3}\right] \leq e^{-O(\mathbb{E}[Z_x]^{1/3})} = n^{-O(\log n)}$$

Furthermore, due to our assumption on $p, q = \Theta(1)$ and the gap $p - q \gg \frac{1}{\log n}$ we obtain

$$\mathbb{E}[Z_x] + \mathbb{E}[Z_x]^{2/3} \ll \mathbb{E}[Y_x] - \mathbb{E}[Y_x]^{2/3}.$$

The above inequalities suggest that the number of 2-paths between $j$ and any vertex $x \in S$ such that $\Psi(x) = \Psi(j)$ is significantly larger compared to the respective count between $j, x'$ such that $\Psi(x') \neq \Psi(j)$ *whp*. Let $\mathcal{B}_j$ be the even that $j$ is misclassified. Combining the above inequalities with Lemma 4 results in

$$\mathbf{Pr}[\mathcal{B}_j] \leq \mathbf{Pr}\left[\bar{\mathcal{E}}\right] + \mathbf{Pr}[\mathcal{B}_j|\mathcal{E}]\,\mathbf{Pr}[\mathcal{E}] = o(n^{-1}).$$

By a union bound over $O(n)$ vertices, the proof is complete.

Finally, the algorithm can be implemented in $\tilde{O}(n)$ time in expectation. Sampling $\tilde{O}(1)$ samples can be implemented in expected $\tilde{O}(1)$ time time, c.f. [17, 35]. Also, checking whether a neighbor of an incoming vertex resides in a given machine can be done in $O(1)$ time by using appropriate data structures to store the information within each machine. □

It is worth noticing that our algorithm is a sublinear time algorithm as the number of edges in $G$ is $O(n^2)$.

### 3.5 Path-$t$ classification

We conclude this section by discussing the effect of the length of the walk $t \geq 2$. Intuitively, $t$ should not be too large, otherwise the random walk will mix. We argue, that among all constant lengths $t \geq 2$, the choice $t = 2$ allows the smallest possible gap for which we can find the true partition $\Psi$. It is worth outlining that $t = 2$ is also in favor of the graph partitioning efficiency as well, since the smaller $t$ is, the less operations are required. Our results extend the results of Zhou and Woodruff [37] to the streaming setting. Our main theoretical result is the following theorem.

Theorem 5. *Let $t \geq 2, t = \Theta(1)$ be the length of a walk. If $p, q = \Theta(1)$ such that $p(1-p), q(1-q) = \Theta(1)$, then $t = 2$ results in the largest possible gap $p - q$ for which we can decide whether $\Psi(u) = \Psi(v)$ or not whp, where $u \neq v \in [n]$.*

To prove Theorem 5, we compute first the expected number of walks of length $t$ between any two vertices in $G(n, k, p, q)$. Because of the special structure of the graph, we are able to derive an exact formula.

Lemma 6. *Let $G \sim G(nk, k, p, q)$ and $p_t = A^t_{uv}, q_t = A^t_{uw}$ be the two types of entries that appear in $A^t$ depending on whether $\Psi(u) = \Psi(v)$ and $\Psi(u) \neq \Psi(w)$ respectively. Then*

$$p_t = (k-1)\frac{n^{t-1}(p-q)^t}{k} + \frac{n^{t-1}(p+(k-1)q)^t}{k},$$

*and*

$$q_t = -\frac{n^{t-1}(p-q)^t}{k} + \frac{n^{t-1}(p+(k-1)q)^t}{k}.$$

Proof. Let $A$ be the $(p,q)$-adjacency matrix defined as $A_{uv} = p$ if $\Psi(u) = \Psi(v)$ and $A_{uw} = q$ if $\Psi(u) \neq \Psi(w)$ for each $u \neq v \in [n]$. It is easy to check that for any $t \geq 1$ the block structure of the planted partition is preserved. This implies that for any $t$, matrix $A^t$ has the same block structure as $A$ and therefore there are two types of entries in each row. Let $p_t, q_t$ be these two types of entries in $A^t$. For $t = 1$, let $p_1 = p$ and $q_1 = q$. Then, by considering the multiplication of the $u$-th line of $A^t$ with the $v$-th column of $A$ we obtain

$$p_{t+1} = pnp_t + (k-1)nqq_t,$$

and similarly by considering the multiplication of the $u$-th line of $A^t$ with the $w$-th column of $A$ we obtain

$$q_{t+1} = qnp_t + pnq_t + \frac{k-2}{k}nqq_t.$$

We can write the recurrence in a matrix form.

$$\begin{bmatrix} p_{t+1} \\ q_{t+1} \end{bmatrix} = M \times \begin{bmatrix} p_t \\ q_t \end{bmatrix}$$

where $M = \begin{pmatrix} pn & (k-1)nq \\ qn & pn + \frac{k-2}{k}nq \end{pmatrix}$. By looking the eigendecomposition of $M$, despite the fact that it is not symmetric, we can diagonalize it as

$$M = USU^{-1},$$

where

$$S = \begin{pmatrix} n(p-q) & 0 \\ 0 & n\big(p+(k-1)q\big) \end{pmatrix}$$

and

$$U = \begin{pmatrix} -(k-1) & 1 \\ 1 & 1 \end{pmatrix}.$$

Given the fact

$$M^k = U \begin{pmatrix} \big(n(p-q)\big)^k & 0 \\ 0 & \Big(n\big(p+(k-1)q\big)\Big)^k \end{pmatrix} U^{-1}$$

and simple algebraic manipulations (omitted) we obtain that

$$p_t = (k-1)\frac{n^{t-1}(p-q)^t}{k} + \frac{n^{t-1}(p+(k-1)q)^t}{k},$$

and

$$q_t = -\frac{n^{t-1}(p-q)^t}{k} + \frac{n^{t-1}(p+(k-1)q)^t}{k}.$$

□

Now, we are able to prove Theorem 5.

PROOF THEOREM 5. Let $p_t = A^t_{uv}, q_t = A^t_{uw}$ where $u, v, w \in V(G)$ such that $\Psi(u) = \Psi(v) \neq \Psi(w)$ and $A$ is defined as in Lemma 6. Also, define $\bar{A}$ to be the result of the randomized rounding of $A$. By Lemma 6 we obtain

$$p_t - q_t = \frac{n^{t-1}(p-q)^t}{k^t}.$$

Notice that we substituted $n$ by $n/k$ as we $G$ has $n$ vertices, with exactly $n/k$ vertices per cluster. Now, suppose $|A^t_{uv} - \bar{A}^t_{uv}| \leq \gamma$, where $\gamma > 0$ is large enough such that the inequality holds *whp* and will be decided in the following. Then, if $\Psi(j_1) = \Psi(j_2) = \Psi(u)$ we obtain the following upper bound

$$|\bar{A}^t_{uj_1} - \bar{A}^t_{uj_2}| \leq |A^t_{uj_1} - \bar{A}^t_{uj_1}| + |A^t_{uj_2} - \bar{A}^t_{uj_2}| = 2\gamma.$$

On the other hand if $\Psi(u) = \Psi(j_1) \neq \Psi(j_2)$, given that $|x| = |x + y - y| \leq |y| + |x - y| \rightarrow |x - y| \geq |x| - |y|$, we obtain the following lower bound

$$|\bar{A}^t_{uj_1} - \bar{A}^t_{uj_2}| \geq |A^t_{uj_1} - A^t_{uj_2}| - 2\gamma = \frac{n^{t-1}(p-q)^t}{k^t} - 2\gamma.$$

Therefore, if $\frac{n^{t-1}(p-q)^t}{k^t} - 2\gamma > 2\gamma$, then there exists a signal that allows us to classify the vertex correctly. In order to find $\gamma$ we need to upper-bound the expectation of the non-negative random variable $Z = (\bar{A}^t_{uv} - A^t_{uv})^2$.

Zhou and Woodruff prove $\mathbb{E}[Z] = \Theta(n^{2t-3})$,c.f. Lemma 5, [37]. The proof of this claim is based on algebraic manipulation. It is easy to verify that $\mathbb{E}[Z]$ is dominated by the terms that correspond to two paths of length $t$ which overlap on a single edge. Applying Markov's inequality, see Theorem 1, we obtain

$$\mathbf{Pr}[Z \geq \mathbb{E}[Z] \log n] \leq \frac{1}{\log n} = o(1),$$

This suggests that setting $\gamma = n^{t-3/2}\sqrt{\log n}$ since $|A^t_{uv} - \bar{A}^t_{uv}| \leq n^{t-3/2}\sqrt{\log n}$ *whp*. The gap requirement is

$$\frac{n^{t-1}(p-q)^t}{k^t} > 4n^{t-3/2}\log n \rightarrow p - q = \Omega\Big(\big(\sqrt{\frac{\log n}{n}}\big)^{1/t}\Big).$$

This proves our claim, as for $t = 2$ we obtain the best possible gap.

□

# 4. EXPERIMENTAL RESULTS

## 4.1 Experimental setup

We refer to our method as EGYPT (*E*fficient which stands for *G*raph *P*ar*T*itioning). The experiments were performed on a single machine, with Intel Xeon CPU at 2.83 GHz, 6144KB cache size and and 50GB of main memory. We have implemented EGYPT in both JAVA and MATLAB. The method we use to compare against for the 1-path classification is LWD, c.f. [31], the single streaming graph partitioning method for which we have theoretical insights [30]. The simulation results were obtained using the JAVA code. The data streaming application was implemented in MATLAB.

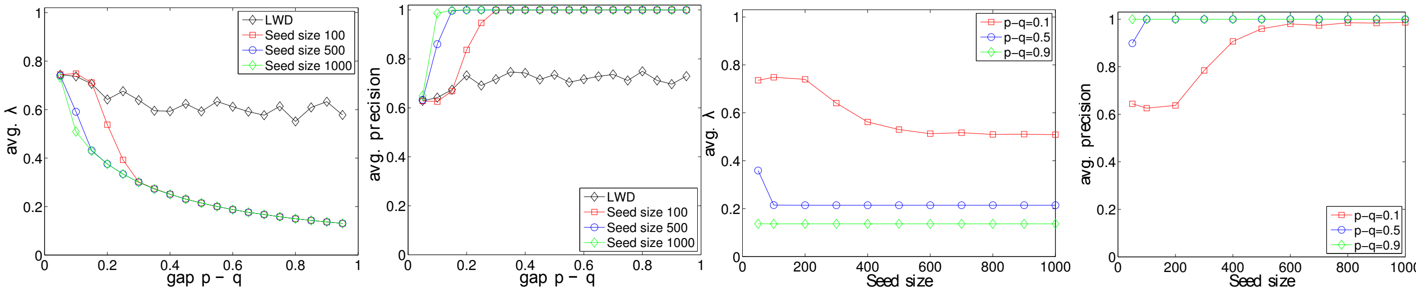
*Synthetic data:* We generate random graphs according to the planted partition model. We fix $q = 0.05$ and we range the gap from 0.05 until 0.95 with a step of 0.05. This results in pairs of $(p, q)$ values with $p$ ranging from 0.1 until 1. We show the results for $n = 8\,000, k = 4$ as the results for other values of $n$ and $k$ ranging from 2 to 16 as successive powers of 2 are qualitatively identical. Notice that given that $p, q = \Theta(1)$ despite the small number of vertices the number of edges is large ranging from 2 to 9.2 million edges. Parameter $B$ ranges in the interval $\{50, 100, 200, \ldots, 1000\}$. The imbalance tolerance $\rho$ was set to 1, demanding equally sized clusters (modulo the remainder of n divided by k). Since groundtruth is available, i.e., function $\Psi$ is known, we measure the precision of the algorithm, as the percentage of the $\binom{n}{2}$ relationships that it guesses correctly. The success of the algorithm is also evaluated in terms of the fraction of edges cut $\lambda$.

*Real data:* An interesting question is what kind of real-world datasets does the planted partition model represent well? Real-world networks typically exhibit skewed degree distributions which are not captured by power-laws. It is worth emphasizing at this point that despite this fact, heuristics that are shown to work provably well using average case analysis can also be successful on real-world data, e.g., [30, 31].

We explore nearest neighbor graphs of well-clustered point clouds. Specifically, we use a perfectly balanced set of 50 000 digits (5 000 digits for each digit 0,1,. . . , 9) from the MNIST database [2]. Each digit is a 28×28 matrix which is converted in a 1-dimensional vector with 784 coordinates. The following numerical evidence indicates that the geometric structure of the dataset is strongly reflected in the nearest neighbor graph. Specifically, we create a 5-nearest neighbor graph and we measure the conductance of the sets induced by each one of the digits. Recall, that the conductance of $U \subseteq V$ the conductance of $U$ is defined as $\phi(U) = \frac{E(U,\bar{U})}{\min(\mathrm{vol}(U),\mathrm{vol}(\bar{U}))}$ where $E(U, \bar{U}) = |\{(u, v) \in E(G) : u \in U, v \in \bar{U}\}|$ and $vol(U) = \sum_{v \in U} deg(v)$. The conductances of the 10 subsets of vertices each corresponding to one of the digits $\{0, \ldots, 9\}$ are 0.0088, 0.0160, 0.0203, 0.0282, 0.0214, 0.0253, 0.0122, 0.0229, 0.0291, 0.0328 respectively.

## 4.2 Simulations

Figures 1(a) and (b) plot the average fraction of edges cut $\lambda$ and the average precision versus the gap $p - q$ for LWD and three runs of EGYPT with different seed sizes, $B \in \{100, 500, 1000\}$. We observe that even for a small value of $B$, the improvement over LWD is significant. Furthermore, we observe that even for $p = 1, q = 0.05$ which




(a) (b) (c) (d)

**Figure 1: (a) Average fraction of edges cut $\lambda$ and (b) average precision versus gap $p-q$ for LWD and EGyPT for three different $B$ values (seed sizes). (c) Average $\lambda$ and (d) average precision, versus $B$ for three different gaps. All data points are averages over five experiments. Observed values are strongly concentrated around their corresponding averages.**

corresponds to the 0.95 gap, LWD is not able to output a good quality partition. This shows that the asymptotic analysis of [30] requires a larger $n$ value in order to recover the partition *whp*. Furthermore, as $B$ increases from 100 to 1000 the quality of the final partition improves. It is worth emphasizing that the results we obtain are strongly concentrated around their corresponding averages. The ratio of the variance over the mean squared was at most 0.0129 and typically of the order $10^{-3}$ indicating a strong concentration according to Chebyshev's inequality, see Theorem 2.

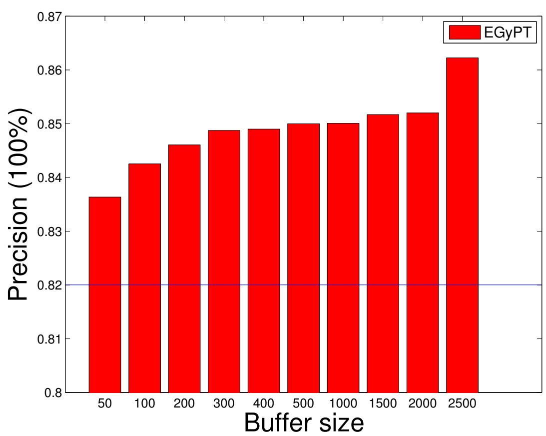



**Figure 2: Digits classification**

Finally, Figures 1(c) and (d) plot the average fraction of edges cut $\lambda$ and the average precision versus the seed size $B$ for three different gaps, averaged over five experiments. Again, data points are concentrated around averages. As expected, the smaller the gap $p-q$ the larger the parameter $B$ has to be in order to obtain a given amount of precision. When the gap is large even for a small seed size $B = 50$, EGyPT obtains the correct partition. Notice that LWD cannot achieve this level of accuracy. Finally, we outline that our method is efficient with respect to run times. Indicatively, we report that for $p = 0.15, q = 0.05$ and $B = 50, 100$ the run times are 0.8 and 1.6 seconds respectively. Similarly for $p = 0.95, q = 0.05$ and $B = 50, 100$ the run times are 2.7 and 8.1 seconds respectively. For these run times, the computational overhead of EGyPT is at the order of $10^{-2}$ seconds.

### 4.3 Streams of data points

We consider a stream of data points where each data point $x \in \mathbb{R}^{784}$ represents a digit from 0 to 9. Whenever a new data point $x$ arrives, we find its $k' = 5$ nearest neighbors among the $B$ first points. This is a variation of the well known $k'$-nearest neighbor graph [22]. Figure 2 shows the improvement in the precision of the clustering as parameter $B$ increases from 0.1% (50) to 5% (2 500) of the total number of data points (50 000). The blue straight line shows the performance of LWD. We observe the improvement over LWD even for $B = 50$ and the monotone increasing behavior of the precision as a function of $B$. While these classification results are not state-of-art in digit classification, they indicate that EGyPT captures the community structure of the underlying graph better than LWD.

## 5. CONCLUSIONS

In this work we introduce the use of higher length walks for streaming graph partitioning, a recent line of research [31] that has already had a significant impact on various graph processing systems. We analyze our proposed algorithm on clustered random graphs.

An interesting research direction is to perform average case analysis using a random graph model with power law degree distribution and small separators, e.g., [10, 11].

## Acknowledgements

The author would like to thank Professor Alan Frieze and Professor Moez Draief and the anonymous reviewers for their feedback.

## 6. REFERENCES

[1] http://tinyurl.com/pmzmtys.
[2] http://yann.lecun.com/exdb/mnist/.
[3] Large scale machine learning and other animals, http://tinyurl.com/qfn766u, June 2014.
[4] B. Bollobás. *Random graphs*. Cambridge University Press, 2001.
[5] A. Buluc, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz. Recent advances in graph partitioning. Arxiv 1311.3144.

[6] R. Chen, J. Shi, Y. Chen, H. Guan, and H. Chen. Powerlyra: Differentiated graph computation and partitioning on skewed graphs. 2013.

[7] A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Structures and Algorithms (RSA)*, 18(2):116–140, 2001.

[8] D. P. Dubhashi and A. Panconesi. *Concentration of measure for the analysis of randomized algorithms*. Cambridge University Press, 2009.

[9] U. Feige and R. Krauthgamer. A polylogarithmic approximation of the minimum bisection. *SIAM J. Comput.*, 31(4):1090–1118, Apr. 2002.

[10] A. D. Flaxman, A. M. Frieze, and J. Vera. A geometric preferential attachment model of networks. *Internet Mathematics*, 3(2):187–205, 2006.

[11] A. Frieze and C. E. Tsourakakis. Some properties of random apollonian networks. In *Internet Mathematics*. 2014.

[12] A. Gionis and C. E. Tsourakakis. Dense subgraph discovery: Kdd 2015 tutorial. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2313–2314. ACM, 2015.

[13] J. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. Powergraph: Distributed graph-parallel computation on natural graphs. In *Operating Systems Design and Implementation (OSDI)*, 2012.

[14] G. Jerrum, M.and Sorkin. The metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, 82(1):155–175, 1998.

[15] U. Kang, C. E. Tsourakakis, and C. Faloutsos. Pegasus: A peta-scale graph mining system. In *IEEE International Conference on Data Mining (ICDM)*, 2009.

[16] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.*, 49(2):291–307, Feb. 1970.

[17] D. E. Knuth. Seminumerical algorithms. 2007.

[18] A. Konstantin and H. Räcke. Balanced graph partitioning. In *ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 120–124, 2004.

[19] R. Krauthgamer, J. S. Naor, and R. Schwartz. Partitioning graphs into balanced components. In *Symposium on Discrete Algorithms (SODA)*, pages 942–949, 2009.

[20] L. Lovász and M. Simonovits. The mixing rate of markov chains, an isoperimetric inequality, and computing the volume. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 346–354. IEEE, 1990.

[21] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Graphlab: A new framework for parallel machine learning. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 340–349, 2010.

[22] M. Maier, M. Hein, and U. von Luxburg. Optimal construction of k-nearest-neighbor graphs for identifying noisy clusters. *Theoretical Computer Science*, 410(19):1749–1764, 2009.

[23] G. Malewicz, M. H. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel: a system for large-scale graph processing. In *SIGMOD '10*, pages 135–146, 2010.

[24] D. Margo and M. Seltzer. A scalable distributed graph partitioner. *Proceedings of the VLDB Endowment*, 8(12), 2015.

[25] F. McSherry. Spectral partitioning of random graphs. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 529–537. IEEE, 2001.

[26] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[27] J. Nishimura and J. Ugander. Restreaming graph partitioning: Simple versatile algorithms for advanced balancing. In *ACM KDD*, 2013.

[28] K. Schloegel, G. Karypis, and V. Kumar. Parallel multilevel algorithms for multi-constraint graph partitioning (distinguished paper). In *Proceedings from the 6th International Euro-Par Conference on Parallel Processing*, Euro-Par '00, pages 296–310, 2000.

[29] D. A. Spielman and S.-H. Teng. A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning. *arXiv preprint arXiv:0809.3232*, 2008.

[30] I. Stanton. Streaming balanced graph partitioning algorithms for random graphs. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1287–1301, 2014.

[31] I. Stanton and G. Kliot. Streaming graph partitioning for large distributed graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012.

[32] C. Tsourakakis. The k-clique densest subgraph problem. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1122–1132. International World Wide Web Conferences Steering Committee, 2015.

[33] C. Tsourakakis, F. Bonchi, A. Gionis, F. Gullo, and M. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 104–112. ACM, 2013.

[34] C. E. Tsourakakis, C. Gkantsidis, B. Radunovic, and M. Vojnovic. Fennel: Streaming graph partitioning for massive distributed graphs. *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014*, 2014.

[35] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller. Triangle sparsifiers. *J. Graph Algorithms Appl.*, 15(6):703–726, 2011.

[36] V. Vu. A simple svd algorithm for finding hidden partitions. *CoRR*, abs/1404.3918, 2014.

[37] H. Zhou and D. Woodruff. Clustering via matrix powering. In *Symposium on Principles of Database Systems (PODS)*, 2004.

# Appendix
## FENNEL is NP-hard

We show that maximizing the optimal quasi-clique objective

$$f_\alpha(\mathcal{P}) = \sum_{i=1}^{k} \Big( e(S_i) - \alpha \binom{|S_i|}{2} \Big),$$

over all possible vertex (disjoint) partitions $\mathcal{P} = (S_1, \ldots, S_k)$ is NP-hard. We reduce the $k$-clique partition NP-hard problem to optimal quasi-clique partitioning: can we partition the vertex set of a graph $G(V, E)$ in $k$ cliques. We claim that if we set $\alpha = 1 - \frac{1}{n^3}$ then the answer to the $k$-clique partitioning problem is YES if and only if $f_\alpha(\mathcal{P}) > 0$. Notice that the latter inequality is equivalent to

$$\sum_{i=1}^{k} \Big( e(S_i) - \binom{|S_i|}{2} \Big) + \frac{1}{n^3} \sum_{i=1}^{k} \binom{|S_i|}{2} > 0.$$

Clearly, if we can partition $V$ in $k$ disjoint cliques $f_\alpha(\mathcal{P}) > 0$. On the other hand, suppose that $f_\alpha(\mathcal{P}) > 0$ for a partition $\mathcal{P}$ that does not correspond to a $k$-clique partitioning of $V$. Then the first summation term $\sum_{i=1}^{k} \Big( e(S_i) - \binom{|S_i|}{2} \Big) \leq -1$ and the second summation term $\frac{1}{n^3} \sum_{i=1}^{k} \binom{|S_i|}{2} < 1$. Hence, we derive a contradiction as $f_\alpha(\mathcal{P}) < 0$.